

# Wordle

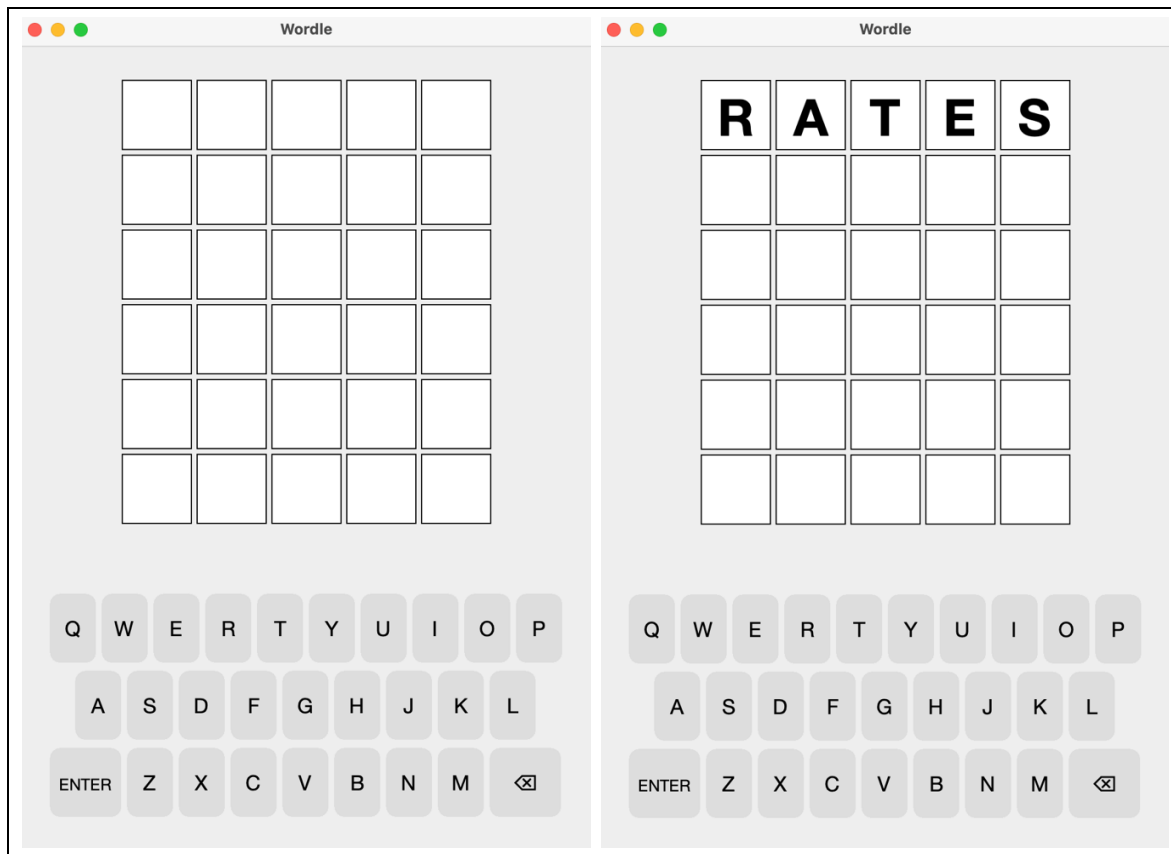
[Acknowledgements to Eric Roberts of Willamette University](#)

This project is designed to help you practice working with strings in the context of an engaging application: the Wordle game initially developed by [Josh Wardle](#), now available on the [New York Times](#) web site. Given Wordle's enormous popularity, we thought it would be fun to give you the chance to implement the game.

To get started on this project, download, extract and save the Wordle project to your computer. The Graphical User Interface (GUI) has been completed for you, so you can run the `main` method in the `Wordle` class to see what it looks like. It creates a window, draws the letter boxes, and creates the keyboard at the bottom of the window. You can even type in letters either by hitting keys on the keyboard or clicking the keys on the screen, just as you can when you are playing the online version. Figure 2, for example, shows both the initial screen and the screen you get after typing in the five letters in the useful starting word `RATES`, which includes five of the most common letters.

Unfortunately, that's all the program does at this point. It doesn't actually let you play the Wordle game. Your job is to make the game playable. But first, it is worth spending a bit of time reviewing the rules for Wordle, in case you've somehow managed to miss this craze.

**Figure 2. Running the starter program**



## Playing Wordle

The object of the Wordle puzzle is to figure out the secret word for the day using no more than six guesses. When you type in a word and then hit the `RETURN` or `ENTER` key, the website gives you information about how close your guess is by coloring the background of the letters. For every letter in your guess that is in its correct position, Wordle colors the background a light shade of green. For every letter that appears in the word but is not in the correct position, Wordle colors the background a brownish yellow. All letters in the guess that don't appear in the word are colored a medium gray.

For example, suppose that the secret word for the day was **RELIC**, and your first guess was **RATES** as in the Figure 2 example. The **R** is in the correct position, and the word contains an **E**, but not in the position you guessed. The secret word does not contain any of the letters **A**, **T**, and **S**. Wordle reports that information by changing the background colors of the squares like this:

**R** **A** **T** **E** **S**

Even though you know the position of the **R**, it doesn't make sense to guess more words beginning with **R** at this point because doing so gives you no new information. Suppose that you tried guessing the word **LINGO**, which contains five new letters, two of which appear in the word, but none of which are correctly positioned. Wordle responds by coloring the letter squares in your second guess as follows:

**L** **I** **N** **G** **O**

Putting these two clues together means that you know that the word begins with an **R**, contains the letters **E**, **L**, and **I** in some order other than the one you guessed, and that the letters **A**, **T**, **S**, **N**, **G**, and **O** do not appear anywhere in the word. These answers give you an enormous amount of information. If you think carefully about it, you might find the word **RELIC**, which is in fact the only English word that meets these conditions:

**R** **E** **L** **I** **C**

Done in three!

It is worth noting a few other rules and special cases. The secret word and each of your guesses must be a real English word that is five letters long. Each guess will be checked against a list of more than 12,000 valid five-letter words. If you guess a word that is not in the word list, Wordle displays a message to that effect, at which point you can delete the letters you've entered and try again. Another rule is that you only get six guesses. If all the letters don't match by then, Wordle gives up on you and tells you what the secret word was.

The most interesting special cases arise when the secret word and the guesses contain duplicates of the same letter. Suppose, for example, that the secret word is **GLASS** and you for some reason guess **SASSY**. Wordle responds with the following colors:

**S** **A** **S** **S** **Y**

The green **S** shows that there is an **S** in the fourth position, and the yellow **S** shows that a second **S** appears somewhere else in the secret word. The **S** in the middle of **SASSY**, however, remains gray because the secret word does not contain three instances of the letter **S**. Properly handling special cases like this are the most challenging aspect of this project.

## The WordleGUI class

The WordleGUI class is already instantiated as the variable `wordle` when you run the main method in Wordle. Even though you don't have to make any changes to WordleGUI or understand the details of its operation, you need to know what capabilities it has on offer so that you can use those facilities in your code. The methods the WordleGUI class are outlined below in Figure 3. The right column of the table gives only a brief description of what these methods do. More complete descriptions appear later in this handout in the description of the step that requires them.

**Figure 3. Useful methods from the wordleGUI class**

<code>wordle.getCurrentRow()</code>	Returns the current row.
<code>wordle.setCurrentRow(row)</code>	Sets the row in which typed characters appear.
<code>wordle.getSquareLetter(row, col)</code>	Returns the letter in the specified row and column.
<code>wordle.setSquareColor(row, col, color)</code>	Sets the color of the specified square.
<code>wordle.getSquareColor(row, col)</code>	Returns the color of the specified square.
<code>wordle.setKeyColor(letter, color)</code>	Sets the color of the specified key letter.
<code>wordle.getKeyColor(letter)</code>	Returns the color of the specified key letter.
<code>wordle.showMessage(message)</code>	Shows a message below the squares.

Here are some predefined constant values that will be helpful in completing the required methods.

**Figure 4. Useful constants from the wordleGUI class**

<code>wordle.N_ROWS</code>	Total number of display rows (normally 6)
<code>wordle.N_COLS</code>	Total number of display columns (normally 5)
<code>wordle.MISSING_COLOR</code>	Light gray color
<code>wordle.CORRECT_COLOR</code>	Green color
<code>wordle.PRESENT_COLOR</code>	Brownish yellow color

Now it's time to color the individual letters. While it is easy to pick a color (green, yellow, or gray), the logic for picking the right color for each letter is more complicated than it seems. We're going to tackle it in this fashion, coloring both the guessed word letters, and the corresponding keyboard letters at the bottom of the display, at the same time.

1. First, we'll color all the guessed word letters gray, as if they were missing. Then we won't have to go back later to deal with any missing letters. This color is the predefined constant `wordle.MISSING_COLOR`.
2. Second, we'll check for any correctly placed letters, one at a time, and color those green. This color is the predefined constant `wordle.CORRECT_COLOR`.
3. Third, we'll deal with any misplaced letters (present in the secret word, but in the wrong spot). This is the challenging task, as we have to know how many repeated letters to color yellow (not necessarily all of them). (Go back and look at the **SASSY** example above if this does not make sense to you yet). This color is the predefined constant `wordle.PRESENT_COLOR`.

All three of these steps are executed so quickly that you will not see them change from gray to either green or yellow.

### Step 0: Set all the guessed letters to gray

Complete the `setAllLettersToMissing` method. You'll need to keep in mind the WordleGUI methods and constants listed above. First, store the game's current row, because we'll need to know which row of letters to color in the display. Second, loop through each of the display columns. Then, get an individual letter and color the entry letter and the keyboard letter. Complete the method as follows.

```

/** Set all letters to missing (gray) color */
private void setAllLettersToMissing()
{
    int row = wordle.getCurrentRow(); // get the current row
    for(int col = 0; col < wordle.N_COLS; col++) // loop through each column
    {
        String letter = wordle.getSquareLetter(row, col); // get a letter
        wordle.setSquareColor(row, col, wordle.MISSING_COLOR); // color entry letter
        wordle.setKeyColor(letter, wordle.MISSING_COLOR); // color keyboard letter
    }
}

```

Check your work by running `WordleTest` to see that this step has been completed successfully.

Notice that each guessed letter (and the corresponding keyboard letter below) are now colored gray.

### **Step 1: Find correct letters (and color them green)**

Complete the `findCorrectLetters` method. You want to check each letter in the method parameter `guess` against the instance variable `secretWord`. Get the current row, and then loop through each of the columns (as you did in Step 0), and use the loop variable to get a corresponding single letter in both `guess` and `secretWord` using the `String` method `substring`. For example, the following statement gets a single letter from position `col` in `guess` (do something similar to also get a single corresponding letter in `secretWord`.)

```
String g = guess.substring(col, col + 1);
```

Then, compare the corresponding single letters in `guess` and `secretWord` to see if they are the same. If so, color both the entry letter and keyboard letter green (`wordle.CORRECT_COLOR`);

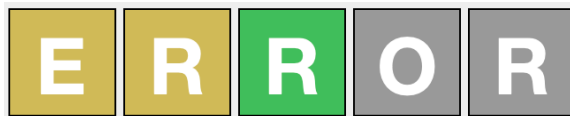
Finally, this method determines if the user guessed the secret word (or not) by returning `true` or `false`. Do this by returning the `boolean` result the comparison of `guess` and `secretWord` (using the appropriate way to compare two `String` objects for equivalence).

Check your work by running `WordleTest` to see that this step has been completed successfully. The tester will display the results of testing your method.

Notice that each correctly guessed letter (and the corresponding keyboard letter below) are now colored green.

### **Step 2: Count the present letters**

“Present” letters are letters that appear in the secret word, but are not in the correct position. The challenge here is for duplicate (or even triplicate) letters. Consider another example (similar to the earlier example). Say that the secret word is `FERRY`, and `guess` is as follows.



Notice that `ERROR` contains three `R`s, while `FERRY` (the secret word) only has two `R`s. The guess above is colored correctly in this case. First the middle `R` is in the correct position so it is colored green. `FERRY` has one other `R`, so the first `R` is colored yellow, but the last `R` is left as is (gray). In other words, before we color any letter yellow, we need to count the number of times that letter appears in the hidden word, and then color that many letters yellow (and no more). Oh, and one more thing; don’t include the green letters in your count.

So, the first method to complete is `presentLetterCount`. This method has one parameter `letter`. Here, we want to count the number of times `letter` appears in `secretWord`, but also check the display to ignore any correctly placed (green) letters. Start by setting up a counter, and then returning the counter at the end of the method. Right after the counter,

set up an integer `col` variable and set it equal to `secretWord.indexOf(letter)`. This method call is going to return the column (integer position) of where `letter` appears in `secretWord`, or -1 if it does not appear at all. Therefore, if `col >= 0` means that `letter` indeed appears in `secretWord`, and we'll loop around again looking for another occurrence of `letter` in `secretWord`. Here is the pseudocode of what needs to be done in this method.

```
get the current row
initialize a counter to zero
initialize a column variable to the position of letter in secret word *
while the column is not negative
    if the display square color is not green // ignore correct (green) letters
        increment the counter
    set column to the position of letter in secret word, starting at position column + 1 **
return the counter
```

```
* using secretWord.indexOf(letter)          ** using secretWord.indexOf(letter, col + 1)
```

Check your work by running `WordleTest` to see that this step has been completed successfully.

### **Step 3: Color the present letters**

Complete the `findPresentLetters` method according to the following pseudocode.

```
get the current row
loop through each letter in guess
    get an individual letter in guess
    initialize count variable to the presentLetterCount(letter)
    initialize a col variable to zero
    while count is greater than zero and col is less than the length of guess
        if letter equals guess character at position col
            if square color(row, col) equals missing color
                set square color(row, col, present color)
            if letter key color equals missing color
                set key color(letter, present color)
            if square color(row, col) does not equal correct color
                decrement count
        increment col
```

Check your work by running `WordleTest` to see that this step has been completed successfully.

Congratulations. You have now completed the Wordle project, and can play the game properly. Feel free to play it as much as you wish. Unlike the New York Times version, you can play this Wordle over and over again.

Please submit a screenshot of the tester success message when you have completed this assignment.